10-04-00

A

SENNIGER, POWERS, LEAVITT & ROEDEL
ATTORNEYS AT LAW

FRANK R. AGOVINO
DERICK E. ALLEN
ROBERT M. BAIN
JAMES J. BARTA, JR.
G. HARLEY BLOSSER
JOHN M. BODENHAUSEN
RICHARD L. BRIDGE
JAMES E. BUTLER, PH.D
SARAH J. CHICKOS
J. BENNETT CLARK
JENNIFER E. COOK
DAVID E. CRAWFORD, JR.
MATTHEW L. CUTLER
JAMES E. DAVIS
KATHRYN J. DOTY
ROBERT M. EVANS, JR.
PAUL I.J. FLEISCHUT
MICHAEL E. GODAR
CHRISTOPHER M. GOFF
DAVID W. HARLAN
EDWARD J. HEJLEK
KAREN Y. HUI
KURT F. JAMES
VINCENT M. KEIL
ANTHONY R. KINNEY
BRIAN P. KLEIN
WILLIAM E. LAHEY
PAUL A. MADDOCK
MICHAEL G. MUNSELL
DEBRA D. NYE
KATHLEEN M. PETRILLO
LAURA R. POLCYN
KEITH A. RABENBERG
STEVEN M. RITCHEY
JOHN K. ROEDEL, JR.
JOSEPH A. SCHAPER
RICHARD A. SCHUTH
MEG MARSHALL THOMAS
MICHAEL J. THOMAS
DONALD W. TUEGEL

ONE METROPOLITAN SQUARE
16TH FLOOR
ST. LOUIS, MISSOURI 63102
314-231-5400

FACSIMILE 314-231-4342
http://www.senniger.com

PATENTS, TRADEMARKS, COPYRIGHTS
AND RELATED MATTERS

OF COUNSEL
IRVING POWERS
DONALD G. LEAVITT
RICHARD G. HEYWOOD

STUART N. SENNIGER
(1921-1997)

October 3, 2000

UTILITY PATENT APPLICATION TRANSMITTAL
(new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket Number: HWH 1825
First Named Inventor: Harry R. Haury
Express Mail Label Number: EL302951713US

TO: Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

APPLICATION ELEMENTS

1. [ ] Fee Transmittal Form
(original and duplicate)

2. [X] Specification [Total Pages 68]

3. [X] Drawings [Total Sheets 6]

4. Oath or Declaration [Total Pages 4]

a. [ ] Newly executed (original or copy)
[ ] New (unexecuted)

b. [X] Copy from a prior application
(for continuation/divisional with
Box 17 completed)
i. [ ] DELETION OF INVENTOR(s)
Signed statement attached
deleting inventor(s) named
in prior application.

5. [X] Incorporation By Reference
(useable if Box 4b is marked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. [ ] Microfiche Computer Program (Appendix)

7. [ ] Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
   a. [ ] Computer Readable Copy
   b. [ ] Paper Copy (identical to computer copy)
   c. [ ] Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. [ ] Assignment Papers (cover sheet & document(s))

9. [ ] 37 CFR 3.73(b) Statement [X] Power of Attorney

10. [ ] English Translation Document (if applicable)

11. [ ] IDS with PTO-1449 [ ] Copies of IDS Citations

12. [ ] Preliminary Amendment

13. [X] Return Receipt Postcard

14. [ ] Small Entity Statement(s)
    [X] Statement filed in prior application; status still proper and desired (copy enclsed)

15. [ ] Certified Copy of Priority Document(s) if foreign priority is claimed

16. [ ] Other: _____

IF A CONTINUING APPLICATION, CHECK APPROPRIATE BOXES AND SUPPLY THE REQUISITE INFORMATION

17. [X] Continuation of prior application No. 08/832,787, filed April 4, 1997, issued October 3, 2000 as U.S. Patent No. 6,128,647, which is a formal application based on U.S. Serial No. 60/014,887, filed April 5, 1996.

    [ ] Complete Application based on provisional Application No. _____.

CORRESPONDENCE ADDRESS

18. Correspondence Address: Customer Number 321
    Attention: Frank R. Agovino

Respectfully submitted,

Frank R Agovino

Frank R. Agovino, Reg. No. 27,416

FRAcwa

Applicant or Patentee: Harry R. Haury Attorney's
Serial or Patent No.: to be assigned Docket No. HWH 1821
Filed or Issued: April 4, 1997
For: Self Configuring Peer to Peer Inter Process Messaging System

STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS (37 CFR 1.9 and 1.27 (b) - INDEPENDENT INVENTOR

As a below named inventor, I hereby declare that I qualify as an independent inventor as defined in 37 CFR 1.9(c) for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, to the Patent and Trademark Office with regard to the invention entitled

SELF CONFIGURING PEER TO PEER INTER PROCESS MESSAGING SYSTEM

described in:

[X] the specification filed herewith
[ ] application serial no. _____, filed _____
[ ] patent no. _____, issued _____

I have not assigned, granted, conveyed or licensed and am under no obligation under contract or law to assign, grant, convey or license, any rights in the invention to any person who could not likewise be classified as an independent inventor under 37 CFR 1.9(c) if that person had made the invention, or to any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

Each person, concern or organization to which I have assigned, granted, conveyed, or licensed or am under an obligation under contract or law to assign, grant, convey, or license any rights in the invention is listed below:

[X] no such person, concern or organization
[ ] persons, concerns or organizations listed below*

*NOTE: Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27).

FULL NAME _____

ADDRESS _____

[ ] INDIVIDUAL [ ] SMALL BUSINESS CONCERN [ ] NONPROFIT ORGANIZATION

FULL NAME _____

ADDRESS _____

[ ] INDIVIDUAL [ ] SMALL BUSINESS CONCERN [ ] NONPROFIT ORGANIZATION

FULL NAME _____

ADDRESS _____

[ ] INDIVIDUAL [ ] SMALL BUSINESS CONCERN [ ] NONPROFIT ORGANIZATION

I acknowledged the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Harry R. Haury		
NAME OF INVENTOR	NAME OF INVENTOR	NAME OF INVENTOR
Harry R Haury		
Signature of Inventor	Signature of Inventor	Signature of Inventor
4/4/97		
Date	Date	Date

SELF CONFIGURING PEER TO PEER INTER PROCESS MESSAGING SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. Serial No. 08/832,787, filed April 4, 1997, now U.S. Patent No. 6,128,647 which claims priority based on provisional application Serial No. 60/014,887, filed April 5, 1996.

BACKGROUND OF THE INVENTION

The traditional limitation of network based inter process messaging and control systems is the incompatibility of the messaging and system control conventions between resources such as various network operating systems and network topologies. With the advent of more ubiquitous networks, significant effort has been expended to enable various operating systems to interact at a basic level by enabling the transfer of data to and from other system environments through the use of compatible data files. The widespread availability of operating system support for data file transfer between incompatible operating environments provides an effective means of automating the transfer of messages and the execution of control instructions between systems that might otherwise be incompatible.

In imaging systems, many vendors have unsuccessfully tried to connect the database directly to the imaging process software across incompatible networks. There is a need for a new operating system independent protocol which does not employ operating system dependent messaging systems such as DDE or OLE and which operates at a higher level so that the protocol deals directly with the process software.

SUMMARY OF THE INVENTION

It is an object of this invention to provide a system and software which allows peer to peer communication and remote process control between processes operating in incompatible operating environments without the need for a master control program.

It is a further object of this invention to provide a messaging protocol which is available to all processes including incompatible processes and which allows each process to read and write files using the protocol.

It is another object of this invention to define a messaging paradigm which is based on file passing technology and which connects various processes through the creation of files.

It is another object of this invention to provide a simple distributed computer environment (SDCE).

It is another object of this invention to provide a computer system that is compatible with a large variety of systems and applications due to the frequency with which other systems can write and copy the text files.

It is another object of this invention to provide a computer system capable of linking incompatible applications and computer systems independent of the computer operating systems being used.

It is another object to provide a system that can move messages and control instructions across an arbitrary number of networks and other connections that allow for the eventual transmission of the messages because of the ease of moving the small ASCII instruction files.

It is still another object of this invention to provide a system which is capable of adding new functions to obsolete and otherwise incompatible legacy systems.

It is another object of this invention to provide an interprocess peer to peer messaging system that can connect any number of processes sequentially or in parallel.

It is an object of this invention to provide an interprocess peer to peer system that uses common virtual or physical disk space on any network with file services to connect resources.

It is an object of this invention to provide an interprocess peer to peer system that allows processes to be stacked by the arbiter so that multiple steps can be performed as a single function, such as read routing, package data and instruction file, encrypt file and copy file.

It is an object of this invention to provide an interprocess peer to peer system that allows processes to be stacked as a result of its intrinsic design and as a result of it being able to execute processes by the arbiter.

In one form, the invention comprises a network system comprising a plurality of resources, some of which being incompatible with others, a network interconnecting the resources and an arbiter resident in each of the resources. The arbiter sends messages via the network and receiving messages via the network. Each arbiter independently reviews and processes the messages from other arbiters of other resources so that the resources communicate directly with each other without the need for a master controlling program and without the need for

other gateway for controlling and processing the messages as the messages are transmitted between resources.

In another form, the invention comprises a message system for transmitting messages on a network between resources interconnected by the network. An arbiter resident in each of the resources sends messages via the network and receives messages via the network, each said arbiter independently reviewing and processing the messages so that the resources communicate directly with each other. As a result, there is no need for a master controlling program or need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

In another form, the invention comprises an inter process peer to peer messaging system for communicating between a plurality of networked resources, some of which employ operating systems which are incompatible with each other. An arbiter message originator associated with each of the resources provides an arbiter message to be sent to the other resources, the arbiter message instructing one of the other resources to execute one or more of the following: remote program execution, data transport, message communication, status communication and relocation of computer resources. A message arbiter receiver associated with each resource receives the arbiter messages from the other resources and responds to the received arbiter message by executing one or more of the following: retransmitting the arbiter message to another one of the resources; and interpreting and executing the received arbiter message wherein the arbiter message originator and the arbiter message receiver do the actual communication between their respective resources without the need for a master

controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

In another form, the invention comprises an inter process peer to peer messaging process for communicating between a plurality of networked resources, some of which employ operating systems which are incompatible with each other. The process comprising the steps of:

transmitting an arbiter message from one resource to the other resources, the arbiter message instructing one of the other resources to execute one or more of the following: remote program execution, data transport, message communication, status communication and relocation of computer resources; and

receiving the arbiter messages from the other resources and for responding to the received arbiter message by executing one or more of the following: retransmitting the arbiter message to another one of the resources; and interpreting and executing the received arbiter message wherein the actual communication between their respective resources is accomplished without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

Other objects and features will be in part apparent and in part pointed out hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a logic diagram of the inter process messaging system according to one preferred embodiment of the invention.

Figure 2 is a flow chart of the logic steps performed by an arbiter of the inter process messaging and control system according to one preferred embodiment of the invention. Appendix A is a Visual Basic source code listing of the arbiter of the invention.

Figure 3 illustrates the basic content of context defined and content defined messages transferred between different, incompatible applications which are linked by the inter process messaging system according to one preferred embodiment of the invention.

Figure 4 describes in detail the mechanism used by the master routing arbiter and any other arbiters to dynamically build routing tables in order to determine how to move a control message from one process to another. An example of an arbiter that uses fixed routing to move messages from one scratch space to another is contained in Figure 4A of Appendix A.

Figure 5 describes the nature of the Ping message that initially is sent to all locations and is used to establish routing on the network. The Ping message content is fully explained in this figure. After receipt of the Ping message, each arbiter sends out its own unique table identifying itself and the originating resource sending the Ping message takes an inventory of all the arbiters which send out tables in response to the Ping message.

Figure 6 describes a number of special pre-registered instructions for network commands that are directly executed by the arbiter. Contextual arbiters use fixed pre-registered commands. An example of such an arbiter is contained in Figure 3A of Appendix A.

Figure 7 is a functional block diagram illustrating an image enabling process according to the invention on a

stand alone personal computer using a context defined simplified distributed computing environment (SDCE) and a local arbiter.

Corresponding reference characters indicate corresponding parts throughout the drawings.

BRIEF DESCRIPTION OF THE APPENDIX

Appendix A, figure 1A illustrates source code for an identity file.

Appendix A, figure 2A illustrates source code for a multi-step communication configuration requiring multiple message files to complete the instruction sequence.

Appendix A, figure 3A illustrates source code for implementing an arbiter process based on contextual file content.

Appendix A, figure 3B illustrates source code for implementing an arbiter process based on contentual file content.

Appendix A, figure 4A illustrates a source code program listing in Visual basic of a message replicating arbiter that uses fixed routing to move messages from one scratch space to another.

DETAILED DESCRIPTION OF THE INVENTION

The system of the invention uses a structured process for object and token passing with an ability to dynamically build network routes between processes resident on different or the same computer, i.e., peer to peer. The processes may be compatible, partially compatible or incompatible. The invention enables the directed transfer of message files from one process to another process without having to have the originating process know the location of another process in a

heterogeneous networking environment; the messages are transferred between process names not locations. The use of independent network arbiter agents, one at each resource or group of resources, to copy and interpret control files allows for a very sophisticated remote peer to peer and/or process to process communications and control. The use of a file based message paradigm according to the invention rather than a set of memory or operating system based variables, provides for much greater flexibility to connect otherwise incompatible systems than would be allowed by other network operating system or process specific messaging systems. The simplicity of writing files of the invention also makes it much more convenient to incorporate this interprocess communication and control into a network of separate systems developed in incompatible operating environments. This is due to the ease with which the messages can be copied between the source and target systems. The use of the file based messaging system of the invention also allows obsolete legacy systems to communicate across a system with minimal programming.

The invention comprises means for automatically sending data, messages, and control instructions between processes operating in a single computer or across a complex heterogeneous network environment. The system was designed to write, optionally encrypt, copy, transmit, interpret and execute instructions, and move data based on instructions contained in small, simple to create files. Each process sends messages to an arbiter which has a resource list of all objects which can be executed. Arbiters may be general purpose or have a specific function such as a replicator arbiter or one-sided (end node) arbiter.

In general, an arbiter according to the invention is an independent process which reviews a message including instructions and processes the instructions of the message when the arbiter determines that the message has a token or address which matches the address of the resource with which it is associated or resident. Traditionally, network messaging systems have a master control program (MCP) for controlling messages between resources connected to the network. For example, centralized e-mail would be such a system. All messages passed through the master control program which acts as a gate keeper for maintaining timing, routing and compatibility. Therefore, the master control program is necessary to allow communication between resources. The invention uses arbiters which avoid the need for such intervention by a master control program and essentially avoid the need for a gate keeper. Arbiters communicate with each other by reading and writing the messages to designated scratch spaces.

In general, traditional networks reach a point of difficulty adding additional users or resources because of their complexity and the concentrated loading which occurs as additional users or resources increase. On the other hand, the invention allows the addition of resources without any increase in complexity. An additional resource may be added simply by providing a unique address to it and by having an arbiter which is able to determine or otherwise know its address. All other arbiters would then be advised of the new address of that resource in order to be able to communicate with the resource. In other words, the arbiters of the invention provide a gateway function between resources

without the need for intervention of the type that the master control program requires.

The message file from an originating arbiter acts as a token which identifies which process is in control of an arbiter at any particular time. The token also controls permission to read and execute the file. In other words, the message functions as a token that passes control between particular arbiters. First, the destination arbiter verifies that the token is part of a valid message and then the destination arbiter gives this message control by allowing the instructions associated with the token to take exclusive control of the destination arbiter. When a destination arbiter receives a message or token, it temporarily turns its message receiving subsystem off so that it does not receive other conflicting messages or tokens. After completing the analysis and/or execution of a message, the destination arbiter then is ready for the next message. The token is part of a larger message which includes logic embedded therein which instructs the destination arbiter to operate in a particular way and utilize the appropriate resources. In other words, the message is a structured object and can have two forms: contextually defined messages and content messages as illustrated in Figure 3. Contextually defined messages include content messages but also include inferred instructions interpreted from the message name. The content message includes lines of code which define a particular action and the way to execute it. The message, including a token, may be considered a virtual file. The name of the contextually defined message is essentially the definition of the destination of the message. The extension name defines the action to be taken.

In one preferred embodiment of the invention, the invention may include a multi-step communication configuration requiring multiple message files to complete the instruction sequence. An example can be seen as implemented as part of the source code included herewith. For example, see Figure 2A, beginning at line 15.

One unique feature of the invention is that it is not specific to a particular operating system but provides a messaging paradigm which can be used with any type of operating system. As a result, the invention may be used to allow a mainframe to communicate with a desktop without the need for a master control program. In general, the system of the invention allows an interface between different, incompatible systems. For example, the invention has been used to allow a mainframe to drive a Windows program on a desktop computer without the need for DDE or OLE messaging systems, which are specific to the Windows operating system, to be resident in the mainframe.

In one aspect of the invention, the system of the invention may be used to allow imaging between incompatible systems. For example, a Sun computer may scan a document and then provide a message to a PC which can then access the scanned document and print it. Most, if not all systems, can read and write ASCII text files, so that it becomes clear that it is easily possible to configure a network of incompatible computers which can access each other's ASCII files by using the arbiter of the invention. Such a system would not require a translator or other master control program which would tend to complicate and slow down the communication between programs.

As noted above and illustrated in Figure 7, it is also contemplated that encryption may be used so that each message is encrypted to further enhance the security features of the network.

In summary, the invention comprises a network comprising a plurality of resources or applications, some of which being incompatible with others. The network interconnecting the resources. An arbiter resident in each of the resources sends messages as a message originator via the network and receives messages as a message arbiter 106 via the network wherein each arbiter independently reviews and processes the messages from other arbiters of other resources so that the resources communicate directly with each other without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources. The network includes a distributed computing environment interconnecting systems using different operating systems and networking systems and wherein the arbiter message comprises ASCII text files as illustrated in figure 3 for the transmission of instructions between resources. In particular, the arbiter message comprises independent task arbiters 100, 106 operating across the network that can dynamically interact with other task arbiters without the need for a central master control system. The independent task arbiters are independent network agents acting under the sole control of the messages being received. The resources communicate directly with other resources via the independent task arbiters without the intervention of any other process. The independent message arbiters provide asynchronous messaging between resources of the network so that each generated message

is transmitted through the network independent of any other messages and the transmitted message will be acted on as it is received by the destination resource. The originating resource executes other tasks after transmitting the message arbiter thereby creating an intrinsically multi-tasking and multi-threaded control system such that multiple arbiter messages can be transmitted through the network independently between multiple resources. The destination arbiter determines whether any necessary data or programs are available for executing the controlled process and writes a control file instructing other network arbiters to transmit the necessary data or programs to the destination arbiter when the destination arbiter determines that the necessary data or programs are not available. Each arbiter employs messages which may be encrypted so that the network is substantially secure.

The arbiter messages include text interconnecting resources across a network or interconnecting resources within a single computer. Each resource processes the arbiter messages in its background while performing other functions in its foreground. Adding other computer program functions is accomplished by executing arbiter instruction files in the background so that programs that provide additional functions can be executed by other resources that can write the instruction files whereby this execution can be so tightly bound that the executed programs appear to be part of the originating program. Data and software are remotely distributed by directly controlling linked computer systems so that executed programs can do such things as copy files to remote locations. The network may handle time independent instructions and the arbiters may be programmed to

execute only at certain times and the programs themselves can be programmed to execute at specific times or intervals by the resources whereby network traffic can be controlled to minimize traffic volume or processor requirements at particular times. The arbiter includes a message replicating arbiter that uses fixed routing to move messages from one scratch space to another.

In another form, the invention comprises a message system for transmitting messages on a network between resources interconnected by the network. An arbiter resident in each of the resources sends messages via the network and receives messages via the network, each said arbiter independently reviewing and processing the messages so that the resources communicate directly with each other. As a result, there is no need for a master controlling program or need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

In another form, the invention comprises an inter process peer to peer messaging system for communicating between a plurality of networked resources, some of which employ operating systems which are incompatible with each other. An arbiter message originator associated with each of the resources provides an arbiter message to be sent to the other resources, the arbiter message instructing one of the other resources to execute one or more of the following: remote program execution, data transport, message communication, status communication and relocation of computer resources. A message arbiter receiver associated with each resource receives the arbiter messages from the other resources and responds to the received arbiter message by executing one or more of the following: retransmitting the arbiter message to

another one of the resources; and interpreting and executing the received arbiter message wherein the arbiter message originator and the arbiter message receiver do the actual communication between their respective resources without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

In another form, the invention comprises an inter process peer to peer messaging process for communicating between a plurality of networked resources, some of which employ operating systems which are incompatible with each other. The process comprising the steps of:

transmitting an arbiter message from one resource to the other resources, the arbiter message instructing one of the other resources to execute one or more of the following: remote program execution, data transport, message communication, status communication and relocation of computer resources; and

receiving the arbiter messages from the other resources and for responding to the received arbiter message by executing one or more of the following: retransmitting the arbiter message to another one of the resources; and interpreting and executing the received arbiter message wherein the actual communication between their respective resources is accomplished without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

Preferably, the process of the invention includes resources which originate messages of simple ASCII text files and wherein the resources identify the system

identity of messages from the text file. The text files may contain a digital signature such as illustrated at line 12 of Appendix A to prevent unauthorized tampering and/or to verify the message source.

Coding Details

No particular programming language, computer system, or network operating system have been indicated for carrying out the various procedures described above. This is in fact due to the broad applicability of this invention to many computer languages, computer systems and network operating systems. Computers which may be used in practicing the invention are diverse and numerous. It is considered that the operations and procedures described as part of the invention are sufficiently disclosed to permit one of ordinary skill in the art to practice the instant invention. One preferred embodiment of implementation of the invention in Visual Basic source code is found below as Appendix A.

Description of Preferred Embodiments
Illustrated in the Figures

Figure 1 shows a diagram that demonstrates the basic logic of the inter process messaging system embodied by the invention. The process originating the message first configures its identity by reading its local identity file. In particular, a resource employs a message originator 100 to read an originator ID 102. The identity file establishes several defaults for communication, the name of the process, and the location of the scratch space (e.g., RAM disk) to which the process is bound. The originating process writes its

message files (described below) which contain its control instructions written with optional encryption in a local storage area that is set in the identity file and is known as a scratch space binding area 104. This area can be used by multiple processes to send messages back and forth. A special process called a message arbiter 106 monitors this scratch space binding area for each new instruction file. When a new file is written, all arbiters monitoring this scratch space read it including a destination arbiter which is identified by the file. After the destination arbiter reads it, the destination arbiter recognizes the file as including the destination arbiter's address or token. As a result, the destination arbiter interprets what should be done and executes the instructions contained within the file. In Figure 1, message arbiter 106 executes a control process. The instructions may consist of one or more of the following: message replication in at least one other scratch space, execution of instructions, launching of new processes, erasing old message files, or requesting that programs or data be copied so that they may be used or analyzed. The process is inherently asynchronous and, without direct connection between processes, the network of arbiters handle message transmission and instruction execution. However, synchronous type behavior can be programmed into the system by providing for a general result of the process 110 and an optional process confirmation 112 so that execution and error conditions are returned to the original process. An example identity file is contained in Figure 1A of Appendix A. Code demonstrating the writing of a message file and confirming instruction file as it would be written by a process using the messaging

system is demonstrated in Figure 2A of Appendix A, beginning at line 21.

Figure 2 describes the logic embedded in the invention to process and interpret these message files. By transferring and executing messages on a peer to peer basis, the invention can be used to distribute data and software, remotely execute programs and procedures, link heterogeneous systems, display images, play multimedia and sound clips, and develop distributed computing applications. Process A is the message originator 100 which may be any external process (e.g., database application or spreadsheet) that will communicate with another external process (control process 108) via the arbiter. Process A begins execution of itself at step 202. At step 204, process A reads its local identity file 102. At step 206, process A writes control instructions for the message arbiter 106 of controlled process 108. These instructions are written to the scratch space binding area 104 defined in the local identity file which is the originator ID 102 of the message originator 100. At step 208, the message arbiter 106 reads the message header in the binding area 104 to determine the controlled process 108 to which the message will be sent. Optionally, the message may be encrypted. At step 210, message copies are transferred by the arbiters from the scratch space of origin to the destination scratch space through an arbitrary number of connected scratch space areas. Arbiters connecting scratch spaces copy original messages and destroy the actual original. At step 212, the arbiter executes the controlled process 108. In particular, the general result of process 110 is that the message is read by the destination arbiter which executes the controlled

process. This final destination arbiter interprets the instruction set. The destination arbiter also determines if required data and programs are inaccessible to the destination process. In particular, at step 214, the destination arbiter determines whether the data sets and programs necessary to execute the controlled process are available. If the necessary data sets and programs are available, the destination arbiter proceeds to step 216 and executes the controlled process. If the necessary data sets and programs are not available, the destination arbiter proceeds to step 218 and writes a control file requesting the data and/or programs needed for executing the controlled process. After the data sets or programs arrive at step 220, the destination arbiter proceeds to step 216 and executes the controlled process. In summary, the destination arbiter determines whether any necessary data or programs are available for executing the controlled process and writes a control file instructing other network arbiters to transmit the necessary data or programs to the destination arbiter when the destination arbiter determines that the necessary data or programs are not available.

Figure 3 describes the basic content of the messages using two different implementations of the invention. Context defined messages (A) are easy to construct and control but they do not have the flexibility and power of the content defined messages (B). The content defined messages (B) must at a minimum have a source ID which identifies the originating process and a destination ID that tells where the message is to be sent. Data set and program lines in the message file identify the data and programs needed to execute the instructions. Special instructions are programs that have been given registered

aliases to simplify using them. Keyboard execution provides a means of sending keystrokes to an application that is launched by the arbiter. Confirmation request instructs the destination arbiter to send two messages back to the originating arbiter; message received and a message regarding the success of program execution when the launched programs are finished running.

An example of code implementing an arbiter process based on contextual file content is contained in Figure 3A of Appendix A, beginning at line 72. Figure 3A demonstrates the contents of a context defined message file. The destination arbiter is defined by the filename. The controlled process is defined by the file extension. The data sets to be used by the controlled process are defined by file contents.

Figure 3B also demonstrates the contents of a content defined message file. An example of a code routine implementing an arbiter process based on contentual file content is contained in Figure 3B of Appendix A, beginning at line 374. This content code routine reads and executes content defined messages and is used as an alternate to the context code routine in Figure 3A of Appendix A for reading context defined messages. In other words, the content code routine of Figure 3B of Appendix A executes content based instruction files whereas the context code routine of Figure 3A of Appendix A executes context based instruction files. The system determines whether the instruction files are content or context based and applies the appropriate routine. The destination arbiter is defined by the filename. The order to the lines in not critical. The line beginning "Source ID:" specifies the identity of the process that originated the message.

The line beginning "Destination ID:" specifies the identity of the destination arbiter. Any line(s) beginning "Data Set:" specifies the file(s) that contain the data necessary to execute the controlled process. Anyn line(s) beginning "Program:" specifies the controlled process program(s) to be executed. Any line(s) beginning "Special Instructions:" specifies any special instructions required for the control process command line. Any line(s) beginning "Keyboard Execution:" specifies redirected key inputs when the arbiter acts as a keyboard robot for the controlled process. A line beginning "Confirmation Request:" specifies whether the controlled process is to send an acknowledgement message of the its execution of the message to the controlled process. A line beginning "Return ID:" specifies the identity of a secondary destination arbiter to which the output of the controlled process will be sent. A line beginning "Return Data Set:" specifies the file that will contain the output of the controlled process. A line beginning "Return Encryption Level:" specifies the type of security to be implemented in the return message, if any. A line beginning "Network Control:" specifies one or more of the following: the public key of the arbiter or process originating the message; the type of security implemented by the originating arbiter or program; and/or miscellaneous header information. The Date and Time lines specify date and time of the original message. The line beginning with "Sequence Number:" specifies an arbitrary index identifying the order in which messages were originated by the a specific process or arbiter so multiple messages can be sent by a particular process or

arbiter to another process or arbiter without ambiguity of order.

Figure 4 describes in detail the mechanism used by a routing arbiter and any other arbiter to dynamically build routing tables in order to determine how an originating arbiter moves a control message from an originating process through various routing arbiters to a destination arbiter associated with the destination process. An example of an arbiter that uses fixed routing to move messages from one scratch space to another is contained in Figure 4A of Appendix A, beginning at line 626. Initially, at step 402, special ping instructions are written in all connected scratch space binding areas 104 by the routing arbiter. Decision block 404 represents a step or series of steps to confirm that all connected arbiters have read the ping message. These steps also determine if other scratch spaces are connected. If spaces are identified through which the ping message has not passed through before, the process proceeds to step 406 wherein the ping message is written to the other scratch spaces. The process then returns to decision block 402. If no spaces are identified so that all have received the ping message, the process proceeds to step 408 wherein a ping return message is sent back to the routing arbiter by the receiving arbiter and the receiving arbiter stores the route information to its master file. Finally, at step 410, the routing arbiter or source builds net tables of connected arbiters to continue with the actual transfer of instruction files.

Figure 5 describes the nature of the Ping message that is used to establish routing on the network as described in Figure 4. The Ping message content is fully explained in this figure.

Figure 6 describes a number of special pre-registered instructions for network commands that are directly executed by the arbiter. Contextual arbiters use fixed pre-registered commands. An example of such an arbiter is contained in Figure 3A of Appendix A, beginning at line 72.

There are five layers of message encryption used to protect network security: 1) none; 2) message check digits using originating process digital signature (DS); 3) encryption of whole message using destination ID and originating process; 4) encryption of instructions using destination ID and originating process; and 5) encryption based on destination public key. Optionally, the message can be left as unencrypted ASCII; the message can be given a set of check digits using the originators identity as an encryption key; the message can be encrypted using the network digital signature (DS); the body of the message can be encrypted using the destination's digital signature and the originator's digital signature; and the message can be encrypted using only the destinations public key. The encryption processes are arbitrary in nature and can use such techniques as digital signature, public and private key encryption. An example of these algorithms would be the licensed RSA encryption techniques.

Figure 7 is a functional block diagram illustrating an image enabling process according to the invention on a stand alone personal computer using a context defined simplified distributed computing environment (SDCE) message and a local arbiter. In general, figure 7 illustrates the use of a text file via an arbiter to control a process such as imaging or file viewing. The application may be any application such as a database

program running on the personal computer. At step 701, the application reads a user keystroke which has been defined to execute the function of viewing a document image. At step 702, the application writes an SDCE instruction file to the scratch space binding area 104. The instruction file is a text file including instruction data such as the file identifier of the file to be imaged. In this case, the instruction sequence also includes a commit file which instructs the arbiter to start the controlled process. At step 703, the application writes the commit file. These first three steps are executed by the application.

The next three steps are executed by the arbiter of the stand alone PC which functions as both the originating arbiter and the destination arbiter in a stand alone system. At step 704, the arbiter scans for a message in the scratch space binding area on a periodic basis and detects the commit file written by step 703. Next, at step 705, the arbiter reads and interprets the instruction file associated with the detected commit file. At step 706, the arbiter executes a view program routine (the controlled process) in response to the detected instructions and passes a data pointer which is part of the instruction file to the view program routine. At step 707, the controlled process, i.e., the view program routine, executes and views the file. At step 708, the user exits and the commit file is erased by the program. At step 709, the application resumes as a result of the commit file being erased.

In view of the above, it will be seen that the several objects of the invention are achieved and other advantageous results attained.

As various changes could be made in the above constructions, products, and methods without departing from the scope of the invention, it is intended that all matter contained in the above description and shown in the accompanying drawings shall be interpreted as illustrative and not in a limiting sense.

APPENDIX A

FIGURE 1A

Example Identity file listing - ASCII text file

```
c:
\nations
NuParadigm Imaging, Inc.  Version hcd3.6bc
c:\nations
c:\nations
testfive
Demonstration Only
 4986719
S
c:\faxtest
```

FIGURE 2A

Example subroutine to write viewing file based on data in external database -- source code in Visual Basic

```
NUP3.FRM - 9
viewall.Show 1

If ik% = 8 Then Exit Sub
ij% = ik%
End If
If ij% = 6 Then
If Data1.Recordset.RecordCount > 100 Then
ij% = MsgBox("Over 200 records selected, please limit search!")
Exit Sub
End If
reccnt = 6
End If
command2.Visible = False
command3.Visible = False
command3d1.Visible = False
command5.Visible = False
command6.Visible = False
command12.Visible = False
command11.Visible = False
timer2.Enabled = True
On Error GoTo 122
Open fp + "\flexindx.vew" For Output As #1
If reccnt <> 6 Then
On Error GoTo 128
Print #1, text13.Text + "\" + Text1.Text + " " + text2.Text
[continue]" " + text3.Text
Close #1
Else
On Error GoTo 128
bookmrk = Data1.Recordset.Bookmark
Data1.Recordset.MoveFirst
```

```
325  Print #1, text13.Text + "\" + Text1.Text + " " + text2.Text + "
" + Program Listing in Visual Basic of a contextually driven arbiter:
text3.Text
If Data1.Recordset.EOF Then GoTo 324
Data1.Recordset.MoveNext
  GoTo 325
324 Close #1
Data1.Recordset.Bookmark = bookmrk
End If
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
Open fp + "\flexindx.sw1" For Output As #2
Print #2, " "
Close #2
126  Exit Sub
122 On Error GoTo 125
MkDir fp
Resume Next
125 Resume 126
128 ij% = MsgBox("Error writing view control file", 48)
 Resume 126
End Sub
```

FIGURE 3A

SDCEARB1.FRM - 1

```
VERSION 2.00
Begin Form Arbiter
   BackColor       =   &H00C0C0C0&
   ClientHeight    =   4155
   ClientLeft      =   300
   ClientTop       =   645
   ClientWidth     =   9600
   FillColor       =   &H00C0C0C0&
   Height          =   4560
   Icon            =   (Icon)
   Left            =   240
   LinkMode        =   1  'Source
   LinkTopic       =   "Form1"
   ScaleHeight     =   4155
   ScaleWidth      =   9600
   Top             =   300
   Width           =   9720
   WindowState     =   2  'Maximized
   Begin Timer Timer1
      Enabled         =   0   'False
      Left            =   8400
      Top             =   1560
   End
   Begin FileListBox File1
      Enabled         =   0   'False
      Height          =   225
      Left            =   120
      Pattern         =   "switch*.dat"
      TabIndex        =   1
      Top             =   4560
      Visible         =   0   'False
```

```
105             Width           =   2175
106         End
107         Begin FileListBox File2
108             BackColor       =   &H00C0C0C0&
109             Enabled         =   0   'False
110             Height          =   420
111             Left            =   120
112             Pattern         =   "*.tif"
113             TabIndex        =   2
114             Top             =   1680
115             Visible         =   0   'False
116             Width           =   2295
117         End
118         Begin Label Label2
119             BackColor       =   &H00C0C0C0&
120             Caption         =   "Copyright by NuParadigm Imaging, Inc.
121     1995
122             Height          =   495
123             Left            =   960
124             TabIndex        =   3
125             Top             =   1080
126             Width           =   6615
127         End
128         Begin Label Label1
129             BackColor       =   &H00C0C0C0&
130             Caption         =   "NuParadigm SDCE Arbiter"
131             FontBold        =   -1  'True
132             FontItalic      =   0   'False
133             FontName        =   "Times New Roman"
134             FontSize        =   22.5
135             FontStrikethru  =   0   'False
136             FontUnderline   =   0   'False
137             Height          =   735
138             Left            =   360
```

```
139       SDCEARB1.FRM - 2

140             TabIndex        =   0
141             Top             =   120
142             Width           =   9015
143          End
144       End
```

```
145     SDCEARB1.FRM - 1
146     Dim hld As Integer
147     Sub Command1_Click ()
148     13334 End Sub
149     Sub Form_Load ()
150     On Error Resume Next
151     loadstate = 0
152     faxscale = 0
153     arbiter.Visible = True
154     For i = 1 To 1000
155     fr = fr + 2
156     Next
157     arbiter.Visible = False
158     ld = 0
159     For i = 0 To 32
160     nindex = i
161     jn(i) = getsystemmetrics(nindex)
162     Next
163     f2wt = screen.Width / screen.TwipsPerPixelX
164     f2ht = screen.Height / screen.TwipsPerPixelY
165     fwt = f2wt / 2
166     fht = f2ht / 2
167     scrrat = fht / fwt
168     ' Read identity file
169     Open "SETINI.DAT" For Input As #7
170     Line Input #7, frstdrv    'Default drive
171     Line Input #7, frstpth    'Default Path
172     Line Input #7, label$     'Registration File Information
173     Line Input #7, apptmp     'temporary application alias
174     Line Input #7, commpath   'Scratch space binding area
175     Line Input #7, username   'Process name
176     Line Input #7, registra   'Registered Owner
177     Line Input #7, code$      'Digital signature on identity file
178     Line Input #7, scancode$  'Default option
179     'Subroutine for verifying digital signature
180     'close of digital signature routine
181     Close #7
182     ij% = DoEvents()
```

```
183         ij% = DoEvents()
184         ij% = DoEvents()
185         file1.Path = commpath
186         fp = frstdrv + frstpth
187         ij% = DoEvents()
```

```
188       SDCEARB1.FRM - 2

189        ij% = DoEvents()
190       'Search scratch space for message file intended for process identity
191       file1.Pattern = username + ".SW*"
192       ij% = DoEvents()
193       file1.Enabled = True
194       ij% = DoEvents()
195       ij% = DoEvents()
196       ij% = DoEvents()
197       ij% = DoEvents()
198       ij% = DoEvents()
199       ij% = DoEvents()
200       ij% = DoEvents()
201       ij% = DoEvents()


202       timer1.Interval = 1000
203       'registration message
204       If registra = "Call (800) 240-0505 to register" Then
205       If Date >= CVDate("January 1, 1998") Then End
206       text1.Text = "This product has not yet been registered and has been
207       provided for evaluation purposes only in accordance with NuParadigm's
208       evaluation agreement.  The product must be registered before it is
209       put into regular use.  The software must also be used in accordance
210       with all the terms and conditions of the NuParadigm license
211       agreement.  If you have any questions please call NuParadigm at
212       (800)240-0505."
213       text1.Visible = True
214       timer1.Interval = 15000
215        arbiter.Visible = True
216       End If
217       timer1.Enabled = True

218       333 End Sub

219       Sub Timer1_Timer ()
220       'routine for interpreting contextually based instruction file
```

```
221       timer1.Interval = 1000
222       arbiter.Visible = False
223       On Error Resume Next
224       timer1.Enabled = False
225       file1.Refresh
226       ij% = DoEvents()
227       ij% = DoEvents()
228       ij% = DoEvents()
229       ij% = DoEvents()
230       ij% = DoEvents()
231       ij% = DoEvents()
232        ij% = DoEvents()
233       ij% = DoEvents()
234       If file1.ListCount > 0 Then
235       ij% = DoEvents()
236       ij% = DoEvents()
237       'execute instruction of external program based on type of
238       confirmation file
239       'viewing process
240       If UCase$(file1.List(0)) = UCase$(username + ".sw1") Then
241       ij% = DoEvents()
242       ij% = DoEvents()
243       op% = Shell(apptmp + "\vewengn.exe", 1)
244       ij% = DoEvents()
245       ij% = DoEvents()
246       'Check for continued module execution
247       While getmoduleusage(op%) > 0
248       ij% = DoEvents()
249       ij% = DoEvents()
250       ij% = DoEvents()
251       Wend
```

SDCEARB1.FRM - 3

```
    ij% = DoEvents()
    GoTo 485
    End If
    ij% = DoEvents()
    ij% = DoEvents()
    'Batch printing process
    If UCase$(file1.List(0)) = UCase$(username + ".sw7") Then
    ij% = DoEvents()
    ij% = DoEvents()
    op% = Shell(apptmp + "\batprn.exe", 1)
    ij% = DoEvents()
    ij% = DoEvents()
    While getmoduleusage(op%) > 0
    ij% = DoEvents()
    ij% = DoEvents()
    ij% = DoEvents()
    Wend
    ij% = DoEvents()
    GoTo 485
    End If
    ij% = DoEvents()
    ij% = DoEvents()
    'Printing of specified list
    If UCase$(file1.List(0)) = UCase$(username + ".sw8") Then
    ij% = DoEvents()
    ij% = DoEvents()
    op% = Shell(apptmp + "\listprn.exe", 1)
    ij% = DoEvents()
    ij% = DoEvents()
    While getmoduleusage(op%) > 0
    ij% = DoEvents()
    ij% = DoEvents()
    ij% = DoEvents()
    Wend
    ij% = DoEvents()
    GoTo 485
```

```
289       End If
290       ij% = DoEvents()
291       ij% = DoEvents()
292       'Document scanning
293       If UCase$(file1.List(0)) = UCase$(username + ".sw3") Then
294       ij% = DoEvents()
295       ij% = DoEvents()
296       If scancode$ = "F" Or scancode$ = "f" Then
297       op% = Shell(apptmp + "\fastscan.exe", 1)
298       Else
299       op% = Shell(apptmp + "\twainscn.exe", 1)
300       End If
301       ij% = DoEvents()
302       ij% = DoEvents()
303       While getmoduleusage(op%) > 0
304       ij% = DoEvents()
305        ij% = DoEvents()
306       ij% = DoEvents()
307       Wend
308       ij% = DoEvents()
309       GoTo 485
310       End If
311       ij% = DoEvents()
312       ij% = DoEvents()
313       'Example of feeding keystrokes to application
314       If UCase$(file1.List(0)) = UCase$(username + ".sw9") Then
315       Kill commpath + "\" + username + ".sw9"
316       ij% = DoEvents()
```

SDCEARB1.FRM - 4

```
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        SendKeys "^{ESC}O~"
        End If
        ij% = DoEvents()
        ij% = DoEvents()
        'Instruction to kill process
        If UCase$(file1.List(0)) = UCase$(username + ".sw0") Then
        Kill commpath + "\" + username + ".sw0"
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        ij% = DoEvents()
        End
        End If
        Else
        On Error GoTo 234
        234 g = 1
        End If
        If flag1 = 1 Then
        flag1 = 0
        SendKeys "^{ESC}O~"
```

```
354         End If
355         ij% = DoEvents()
356         ij% = DoEvents()
357         485 loadstate = 1
358         ij% = DoEvents()
359         ij% = DoEvents()
360         ij% = DoEvents()
361         ij% = DoEvents()
362         ij% = DoEvents()
363         ij% = DoEvents()
364         ij% = DoEvents()
365         ij% = DoEvents()
366         ij% = DoEvents()
367         GoTo 3546
368         ij% = DoEvents()
369         ij% = DoEvents()
370          ij% = DoEvents()
371         3546  timer1.Enabled = True
372         timer1.Interval = 1000
373         End Sub
```

374 Figure 3B

Source Code showing scan of Scratch Space Binding Area for messages in Binding Area set to variable in identity file, "Commpath".

```
375 file1.Path = commpath
376  ij% = DoEvents()
377  ij% = DoEvents()
378 'Username identifies process ID
379 file1.Pattern = username + ".*"
380 ij% = DoEvents()
381 file1.Enabled = True
382 ij% = DoEvents()
383 ij% = DoEvents()
384 ij% = DoEvents()
385 ij% = DoEvents()
386 ij% = DoEvents()
387 ij% = DoEvents()
388 ij% = DoEvents()
389 ij% = DoEvents()

    'Set scan interval according to parameter setting
    timer1.Interval = Scan_time
    timer1.enabled = True
```

Source Code Segment Showing Scan for and Interpretation of Content Defined SDCE Message

```
Sub Timer1_Timer ()
timer1.Enabled = False
arbiter.Visible = False
On Error Resume Next
'Scan for messages
file1.Refresh
ij% = DoEvents()
ij% = DoEvents()
special_instruction = ""
key_inst = ""
src_id = ""
dest_ID = ""
data_set = ""
ext_program = ""
confirm = ""
return_id = ""
ret_data_set = ""
ret_encrypt = ""
net_cntrl = ""
message_date = ""
message_time = ""
seq_number = ""
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
'Do we have a message
If file1.ListCount > 0 Then
ij% = DoEvents()
ij% = DoEvents()
'Execution sequence for content defined message
If UCase$(file1.List(0)) = UCase$(username + ".sdc") Then
jk% = 0
Open file1.path + "\" + file1.List(0) For Input As #1
10 Input #1, sdc_line(jk%)
If EOF(1) Then GoTo 15
jk% = jk% + 1
GoTo 10
15 For i = 1 To jk%
If Left$(sdc_line(i), 10) = "Source ID:" Then
src_id = Right$(sdce_line(i), Len(sdce_line(i) - 10))
End If
If Left$(sdc_line(i), 15) = "Destination ID:" Then
dest_ID = Right$(sdce_line(i), Len(sdce_line(i) - 15))
End If
If Left$(sdc_line(i), 9) = "Data Set:" Then
data_set = Right$(sdce_line(i), Len(sdce_line(i) - 9))
End If
```

```
If Left$(sdc_line(i), 8) = "Program:" Then
ext_program = Right$(sdce_line(i), Len(sdce_line(i) - 8))
End If
If Left$(sdc_line(i), 20) = "Special Instruction:" Then
special_instruction = special_instruction + Right$(sdce_line(i), Len(sdce_line(i) - 20))
End If
If Left$(sdc_line(i), 21) = "Keyboard Instruction:" Then
key_inst = key_inst + Right$(sdce_line(i), Len(sdce_line(i) - 21))
End If
If Left$(sdc_line(i), 21) = "Confirmation Request:" Then
confirm = Right$(sdce_line(i), Len(sdce_line(i) - 21))
End If
If Left$(sdc_line(i), 10) = "Return ID:" Then
return_id = Right$(sdce_line(i), Len(sdce_line(i) - 10))
End If
If Left$(sdc_line(i), 16) = "Return Data Set:" Then
ret_data_set = Right$(sdce_line(i), Len(sdce_line(i) - 16))
End If
If Left$(sdc_line(i), 24) = "Return Encryption Level:" Then
ret_encrypt = Right$(sdce_line(i), Len(sdce_line(i) - 24))
End If
If Left$(sdc_line(i), 16) = "Network Control:" Then
net_cntrl = Right$(sdce_line(i), Len(sdce_line(i) - 16))
End If
If Left$(sdc_line(i), 5) = "Date:" Then
message_date = Right$(sdce_line(i), Len(sdce_line(i) - 5))
End If
If Left$(sdc_line(i), 5) = "Time:" Then
message_time = Right$(sdce_line(i), Len(sdce_line(i) - 5))
End If
If Left$(sdc_line(i), 16) = "Sequence Number:" Then
seq_number = Right$(sdce_line(i), Len(sdce_line(i) - 16))
End If
'Call subroutine for verifying permission registration
ij% = call verify(src_id,message_date,message_time,net_cntrl,ext_program)
If ij% = 2 Then
Kill file1.path + "\" + file1.List(0)
GoTo 200
End If
ij% = DoEvents()
ij% = DoEvents()
'Null from shell string are interpreted as skipped message line
op% = Shell(ext_program + " " + data_set + " "+ confirm + " " + return_id + " " + ret_data_set + " " +
ret_encrypt + " " + special_instruction, 1)
ij% = DoEvents()
SendKeys key_inst
ij% = DoEvents()
While getmoduleusage(op%) > 0
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
Wend
ij% = DoEvents()
ij% = DoEvents()
```

```
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
ij% = DoEvents()
200 timer1.Enabled = True
timer1.Interval = 1000End Sub
```

505 SDCE.BAS - 1

```
506     ' Declare Function GetModuleHandle Lib "Kernel" (ByVal lpModuleName
507     As String) As Int
508     eger
509     ' Declare Sub freemodule Lib "Kernel" (ByVal hModule As Integer)
510     ' Declare Function GetModuleUsage Lib "Kernel" (ByVal hModule As
511     Integer) As Integer
512     ' Declare Sub freelibrary Lib "Kernel" (ByVal hLibModule As Integer)
513     ' ij% = DoEvents()
514     ' ij% = DoEvents()
515     ' ij% = DoEvents()
516     ' ij% = DoEvents()
517     ' ij% = DoEvents()
518     ' ij% = DoEvents()
519     ' ij% = DoEvents()
520     ' ij% = DoEvents()
521     ' imsc% = getmodulehandle("imscan")
522     ' imvb% = getmodulehandle("imvb3")
523     ' imim% = getmodulehandle("imgman2")
524     ' While getmoduleusage(imsc%) > 0
525     ' freemodule (imsc%)
526     ' Wend
527     ' While getmoduleusage(imvb%) > 0
528     ' freemodule (imvb%)
529     ' Wend
530     ' While getmoduleusage(imim%) > 0
531     ' freelibrary (imim%)
532     ' Wend
533     ' stop
534     ' End

535     Global registra As String
536     Global scancode$
537     ' Constants for Error Codes
538     Global APPTMP As String
539     Global scaladjy As Single
```

```
Global scaladjx As Single
Global token As Integer
Global scrrat As Single
Global imgrat As Single
Global f2wid As Integer
Global f2hi As Integer
Global zwt As Integer
Global zht As Integer
Global prntmult As Integer
Global nindex As Integer
Global timeok As Integer
Global frst As Integer
 Global lst As Integer
Global fht As Integer
Global f2ht As Integer
Global fwt As Integer
Global f2wt As Integer
Global faxscale As Integer
Global nopaint As Integer
Global newimage As Integer
 Global zlvl As Integer
Global mx As Integer
Global my As Integer
Global himage As Integer
Global zmst  As Integer
Global jnewt As Integer
Global Const IMG_ERR = 0              ' Indicates an error ocurred
Global Const IMG_OK = 1
```

```
568     SDCE.BAS - 2

569     Global Const IMG_FULL = 0
570     Global Const IMG_INV_FILE = 2    ' unsupported file type
571     Global Const IMG_FILE_ERR = 3    ' error reading from file
572     Global Const IMG_NOFILE = 4      ' file not found
573     Global Const IMG_NOTAVAIL = 5    ' info not available
574     Global Const IMG_NOMEM = 6       ' insufficient memory
575     Global Const IMG_CLOSED = 7      ' image file is closed
576     Global Const IMG_INV_HAND = 8    ' invalid image handle
577     Global Const IMG_NSUPPORT = 9    ' image option not supported
578     Global Const IMG_PROC_ERR = 10   ' error processing image
579     Global Const IMG_PRINT_ERR = 11 ' error printing image
580     Global Const IMG_BAD_PRN = 12    ' printer doesnt support bitmaps
581     Global Const IMG_BAD_SRC = 13    ' invalid src rect specified
582     Global Const IMG_BAD_TYPE = 14   ' tried to use ImgGetDIB on Vector
583     Img


584     Global Const IMG_RENDER_SELF = 1
585     Global Const IMG_PRINT_SELF = 2
586     Global Const IMG_PRNT_VECTOR = 4
587     Global flag1 As Integer
588     Global flag2 As Integer
589     Global flag3 As Integer
590     Global flag4 As Integer
591     Global jn(32) As Integer
592     Global FRSTPTH As String
593     Global FRSTDRV As String
594     Global FP As String
595     Global commpath As String
596     Global username As String
597     Global flags As Long
598     ' Define some types we need to make life easy


599     Type RECTANGLE
600             LEFT As Integer
```

```
601             TOP As Integer
602             right As Integer
603             bottom As Integer
604     End Type


605     Type imageinfo

606             bisize As Long
607             LORGWID As Long
608             lorghi  As Long
609             biplanes As Integer
610             bibitcount As Integer
611              bicompression As Long
612             bisizeimage As Long
613             bixperm As Long
614             biyperm As Long
615             biclrused As Long
616             biclrimportant As Long
617     End Type

618     Declare Function getsystemmetrics Lib "User" (ByVal nindex As
619     Integer) As Integer
620     Declare Function getmoduleusage Lib "Kernel" (ByVal hModule As
621     Integer) As Integer
622     Rem Declare Function imginit Lib "img.dll" () As Integer
```

623 SDCE1.BAS - 1

624 Sub unl_mds ()

625 End Sub

FIGURE 4A -- Program Listing in Visual Basic of message replicating arbiter

```
REP3.FRM - 1

VERSION 2.00
Begin Form Form1
   Caption         =   "Message Replicator "
   ClientHeight    =   6780
   ClientLeft      =   60
   ClientTop       =   360
   ClientWidth     =   5610
   Height          =   7185
   Icon            =   (Icon)
   Left            =   0
   LinkTopic       =   "Form1"
   ScaleHeight     =   6780
   ScaleWidth      =   5610
   Top             =   15
   Width           =   5730
   Begin CheckBox Check3
      Caption         =   "Unzip Zip Files - ie.  *.ZIP"
      Height          =   255
      Left            =   360
      TabIndex        =   19
      Top             =   5160
      Value           =   1  'Checked
      Width           =   2535
   End
   Begin CheckBox Check2
      Caption         =   "Updated Files Only"
      Height          =   375
      Left            =   360
      TabIndex        =   18
      Top             =   4680
      Value           =   1  'Checked
      Width           =   2655
```

```
End
Begin TextBox Text3
   Height          =   375
   Left            =   3120
   TabIndex        =   16
   Text            =   "00:00"
   Top             =   4800
   Width           =   1695
End
Begin CommandButton Command3
   Caption         =   "&Pause"
   Height          =   495
   Left            =   3720
   TabIndex        =   15
   Top             =   5760
   Width           =   1575
End
Begin FileListBox File1
   Height          =   225
   Left            =   5040
    TabIndex        =   14
   Top             =   3840
   Visible         =   0   'False
   Width           =   375
End
Begin TextBox Text2
   Height          =   375
   Left            =   3120
   TabIndex        =   12
   Text            =   "1"
   Top             =   3840
   Width           =   1695
```

```
693         REP3.FRM - 2

694             End
695             Begin Timer Timer1
696                 Interval        =   60000
697                 Left            =   2520
698                 Top             =   3840
699             End
700             Begin CheckBox Check1
701                 Caption         =   "Delete after Copy"
702                 Height          =   255
703                 Left            =   360
704                 TabIndex        =   11
705                 Top             =   4320
706                 Value           =   1   'Checked
707                 Width           =   1935
708             End
709             Begin TextBox Text1
710                 Height          =   375
711                 Left            =   360
712                 TabIndex        =   9
713                 Text            =   "*.*"
714                 Top             =   3840
715                 Width           =   1695
716             End
717             Begin CommandButton Command2
718                 Caption         =   "E&xit"
719                 Height          =   495
720                 Left            =   1920
721                 TabIndex        =   7
722                 Top             =   5760
723                 Width           =   1575
724             End
725             Begin CommandButton Command1
726                 Caption         =   "&Run"
727                 Height          =   495
728                 Left            =   240
729                 TabIndex        =   6
```

```
730             Top             =   5760
731             Width           =   1455
732          End
733          Begin DirListBox Dir2
734             Height          =   2505
735             Left            =   3120
736             TabIndex        =   3
737             Top             =   960
738             Width           =   2175
739          End
740          Begin DirListBox Dir1
741             Height          =   2505
742             Left            =   360
743             TabIndex        =   2
744             Top             =   960
745             Width           =   2175
746           End
747          Begin DriveListBox Drive2
748             Height          =   315
749             Left            =   3120
750             TabIndex        =   1
751             Top             =   480
752             Width           =   2175
753          End
754          Begin DriveListBox Drive1
755             Height          =   315
756             Left            =   360
757             TabIndex        =   0
```

```
758         REP3.FRM - 3

759             Top             =   480
760             Width           =   2175
761          End
762          Begin Label Label5
763             Caption         =   "Scan Start Time            (ie.-14:08 or
764         2:08 PM)"
765             Height          =   495
766             Left            =   3120
767             TabIndex        =   17
768             Top             =   4320
769             Width           =   2175
770          End
771          Begin Label Label4
772             Caption         =   "Scan Rate (Minutes)"
773             Height          =   255
774             Left            =   3120
775             TabIndex        =   13
776             Top             =   3600
777             Width           =   1935
778          End
779          Begin Label Label3
780             Caption         =   "Pattern"
781             Height          =   255
782             Left            =   360
783             TabIndex        =   10
784             Top             =   3600
785             Width           =   1695
786          End
787          Begin Label Label2
788             Caption         =   "Copyright - NuParadigm Imaging, Inc.
789         1995"
790             Height          =   375
791             Left            =   840
792             TabIndex        =   8
793             Top             =   6360
794             Width           =   3855
```

```
795         End
796         Begin Label Label1
797            Caption         =   "Destination"
798            Height          =   255
799            Left            =   3120
800            TabIndex        =   5
801            Top             =   120
802            Width           =   2175
803         End
804         Begin Label Source
805            Caption         =   "Source"
806            Height          =   255
807            Left            =   360
808            TabIndex        =   4
809            Top             =   120
810            Width           =   2175
811         End
812      End
```

```
813         REP3.FRM - 1

814         Dim a$
815         Dim b$
816         Dim c$
817         Dim d$
818         Dim intr As Integer
819         Dim timcnt As Integer
820         Dim strt As Integer



821         Sub its_time ()
822         timcnt = 0
823         file1.Refresh
824         On Error GoTo 10
825         For i = 1 To file1.ListCount
826         If check2.Value = 1 Then
827         On Error Resume Next
828         If (DateValue(FileDateTime(file1.Path + "\" +
829         file1.List(file1.ListCount - i))) = Dat
830         eValue(FileDateTime(dir2.Path + "\" + file1.List(file1.ListCount -
831         i))) And TimeValue
832         (FileDateTime(file1.Path + "\" + file1.List(file1.ListCount -i))) >
833         TimeValue(FileDa
834         teTime(dir2.Path + "\" + file1.List(file1.ListCount - i)))) Or
835         DateValue(FileDateTime
836         (file1.Path + "\" + file1.List(file1.ListCount - i))) >
837         DateValue(FileDateTime(dir2.P
838         ath + "\" + file1.List(file1.ListCount - i))) Then
839         On Error GoTo 10

840         FileCopy file1.Path + "\" + file1.List(file1.ListCount - i),
841         dir2.Path + "\" + file1.
842         List(file1.ListCount - i)
843         If check1.Value = 1 Then
844         Kill file1.Path + "\" + file1.List(file1.ListCount - i)
845         ij% = DoEvents()
846         ij% = DoEvents()
```

```
847         ij% = DoEvents()
848         ij% = DoEvents()
849         ij% = DoEvents()
850         End If
851         If check3.Value = 1 Then
852         ChDir dir2.Path
853         ij% = DoEvents()
854         ij% = DoEvents()
855         ij% = DoEvents()
856         ij% = DoEvents()

857         If StrComp(Right$(dir2.Path + "\" + file1.List(file1.ListCount - i),
858         3), "zip", 1) =
859         0 Then
860         'Automatic execution of external process

861          ij% = Shell("pkunzip -o " + dir2.Path + "\" +
862         file1.List(file1.ListCount - i))
863         ij% = DoEvents()
864         ij% = DoEvents()
865         ij% = DoEvents()
866         ij% = DoEvents()
867         ij% = DoEvents()
868         ij% = DoEvents()
869         ij% = DoEvents()
870         ij% = DoEvents()
871         ij% = DoEvents()
872         ij% = DoEvents()
873         End If
874         End If
875         End If
876         Else
877         FileCopy file1.Path + "\" + file1.List(file1.ListCount - i),
878         dir2.Path + "\" + file1.
879         List(file1.ListCount - i)
880         If check1.Value = 1 Then
881         Kill file1.Path + "\" + file1.List(file1.ListCount - i)
882         ij% = DoEvents()
```

```
883         ij% = DoEvents()
```

```
884         REP3.FRM - 2

885         ij% = DoEvents()
886         ij% = DoEvents()
887         ij% = DoEvents()
888         End If
889         If check3.Value = 1 Then
890         ChDir dir2.Path
891         ij% = DoEvents()
892         ij% = DoEvents()
893         ij% = DoEvents()
894         ij% = DoEvents()

895         If StrComp(Right$(dir2.Path + "\" + file1.List(file1.ListCount - i),
896         3), "zip", 1) =
897         0 Then
898         ij% = Shell("pkunzip -o " + dir2.Path + "\" +
899         file1.List(file1.ListCount - i))
900         ij% = DoEvents()
901         ij% = DoEvents()
902         ij% = DoEvents()
903         ij% = DoEvents()
904         ij% = DoEvents()
905         ij% = DoEvents()
906         ij% = DoEvents()
907         ij% = DoEvents()
908         ij% = DoEvents()
909         ij% = DoEvents()
910         End If
911         End If
912         End If
913         ij% = DoEvents()
914         ij% = DoEvents()
915         ij% = DoEvents()
916         ij% = DoEvents()
917         ij% = DoEvents()
918         ij% = DoEvents()
919         ij% = DoEvents()
```

```
920         40 Next
921         'For i = 1 To file1.listcount
922         'FileCopy file1.path + "\" + file1.list(file1.listcount - i),
923         dir2.path + "\" + file1
924         .list(file1.listcount - i)
925         'Next
926         Exit Sub
927         10 Resume 40

928         End Sub

929         Sub command1_click ()
930         timer1.Enabled = True
931         intr = Int(Val(text2.Text))
932         Open "repini.dat" For Output As #1
933         Print #1, drive1.Drive
934          Print #1, drive2.Drive
935         Print #1, dir1.Path
936         Print #1, dir2.Path
937         Print #1, text1.Text
938         Print #1, check1.Value
939         Print #1, intr
940         Print #1, check2.Value
941         Print #1, check3.Value
942         Print #1, text3.Text


943         Close
944         file1.Path = dir1.Path
945         file1.Pattern = text1.Text
```

```
946         REP3.FRM - 3

947         End Sub

948         Sub Command2_Click ()
949         End
950         End Sub

951         Sub Command3_Click ()
952         timer1.Enabled = False
953         End Sub

954         Sub Drive1_Change ()
955         dir1.Path = drive1.Drive
956         End Sub

957         Sub Drive2_Change ()
958         dir2.Path = drive2.Drive
959         End Sub

960         Sub Form_Load ()
961         timcnt = 0
962         strt = 0
963          On Error GoTo ending
964         'read replicating arbiter identity without digital signature
965         Open "repini.dat" For Input As #1
966         Line Input #1, a$
967         Line Input #1, b$
968         Line Input #1, c$
969         Line Input #1, d$
970         Line Input #1, e$
971         Line Input #1, f$
972         Line Input #1, g$
973         Line Input #1, h$
974         Line Input #1, i$
975         Line Input #1, j$
976         'set default source scratch space
977         drive1.Drive = a$
```

```
978         dir1.Path = c$
979         'set default destination scratch space
980         drive2.Drive = b$
981         dir2.Path = d$
982         'set process identity
983         text1.Text = e$
984         'set other operating parameters
985         check1.Value = Val(f$)
986         check2.Value = Val(h$)
987         check3.Value = Val(i$)
988         text3.Text = j$
989         intr = Val(g$)
990         text2.Text = g$
991         Close #1
992         Call command1_click
993          GoTo 20
994         ending:
995         Close #1
996         Resume 20
997         20 End Sub

998         Sub Timer1_Timer ()
999         'copy message files from source to destination
1000        timcnt = timcnt + 1
1001        On Error GoTo start_anyway
1002        If strt = 1 Then
1003        If timcnt >= intr Then
1004        Call its_time
```

```
1005        REP3.FRM - 4

1006        End If
1007        Else
1008        If Time > TimeValue(text3.Text) - (4 / 60 / 24) And Time <
1009        TimeValue(text3.Text) + (4
1010         / 60 / 24) Then
1011        strt = 1
1012        Call its_time
1013        End If
1014        End If
1015        If timcnt > 24 * 60 Then
1016        Call its_time
1017        strt = 1
1018        End If
1019        GoTo 33
1020        start_anyway:
1021        strt = 1
1022        Resume 33
            33 End Sub
```

I claim:

1. A network system comprising:

a plurality of resources, some of which being incompatible with others;

a network interconnecting the resources;

an arbiter resident in each of the resources for sending messages via the network and for receiving messages via the network wherein each arbiter independently reviews and processes the messages from other arbiters of other resources so that the resources communicate directly with each other without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

2. A network system of claim 1 wherein the network includes a distributed computing environment interconnecting systems using different operating systems and networking systems and wherein the arbiter message comprises ASCII text files for the transmission of instructions between resources.

3. A system of claim 1 wherein the arbiter message comprises independent task arbiters operating across the network that can dynamically interact with other task arbiters without the need for a central master control system, wherein said independent task arbiters are independent network agents acting under the sole control of the messages being received, and wherein said resources communicate directly with other resources via the independent task arbiters without the intervention of any other process.

4. A system of claim 3 wherein the independent message arbiters provide asynchronous messaging between resources of the network so that each generated message is transmitted through the network independent of any other messages and the transmitted message will be acted on as it is received by the destination resource.

5. A system of claim 4 wherein the originating resource executes other tasks after transmitting the message thereby creating an intrinsically multi-tasking and multi-threaded control system such that multiple arbiter messages can be transmitted through the network independently between multiple resources.

6. A system of claim 3 wherein the destination arbiter determines whether any necessary data or programs are available for executing the controlled process and writes a control file instructing other network arbiters to transmit the necessary data or programs to the destination arbiter when the destination arbiter determines that the necessary data or programs are not available.

7. A system of claim 1 wherein each arbiter employs messages which are encrypted after creation at the local resource so that the network is substantially secure.

8. A system of claim 1 wherein the arbiter messages include text that provide instructions for interconnecting resources across a network or interconnecting resources within a single computer.

9. A system of claim 1 wherein each resource processes the arbiter messages in its background while performing other functions in its foreground.

10. A system of claim 9 wherein adding other computer program functions is accomplished by executing arbiter instruction files in the background so that programs that provide additional functions can be executed by arbiters that can write the instruction files whereby this execution can be so tightly bound that the executed programs appear to be part of the originating program.

11. A system of claim 1 further comprising means for providing for the remote distribution of data and software by directly controlling linked computer systems so that executed programs can do such things as copy files to remote locations and combine data and/or program files with execution instructions necessary to process the data files.

12. A system of claim 1 wherein the network handles time independent instructions and wherein the arbiters are programmed to execute only at certain times and the programs themselves can be programmed to execute at specific times or intervals by the resources whereby network traffic can be controlled to minimize traffic volume or processor requirements at particular times.

13. A system of claim 1 wherein the arbiter includes a message replicating arbiter that uses routing information to move messages from one scratch space to another and that determines the routing tree between arbiters.

14. A message system for transmitting messages on a network between resources interconnected by the network, said message system comprising:

an arbiter resident in each of the resources for sending messages via the network and for receiving messages via the network, each said arbiter independently reviewing

and processing the messages so that the resources communicate directly with each other without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

15. An inter process peer to peer messaging system for communicating between a plurality of networked resources, some of which employ operating systems which are incompatible with each other, said system comprising:

an arbiter message originator associated with each of the resources for providing an arbiter message to be sent to the other resources, the arbiter message instructing one of the other resources to execute one or more of the following: remote program execution, data transport, message communication, status communication, arbiter identification, data encryption, message encryption, and relocation of computer resources;

a message arbiter receiver associated with each resource for receiving the arbiter messages from the other resources and for responding to the received arbiter message by executing one or more of the following: retransmitting the arbiter message to another one of the resources; and deciphering, interpreting and executing the received arbiter message wherein the arbiter message originator and the arbiter message receiver do the actual communication between their respective resources without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

16. An inter process peer to peer messaging process for communicating between a plurality of networked resources, some of which employ operating systems which are

incompatible with each other, said process comprising the steps of:

transmitting an arbiter message from one resource to the other resources, the arbiter message instructing one of the other resources to execute one or more of the following: remote program execution, data transport, message communication, status communication, arbiter identification, data encryption and message encryption and relocation of computer resources; and

receiving the arbiter messages from the other resources and for responding to the received arbiter message by executing one or more of the following: retransmitting the arbiter message to another one of the resources; and interpreting and executing the received arbiter message wherein the actual communication between their respective resources is accomplished without the need for a master controlling program and without the need for other gateway for controlling and processing the messages as the messages are transmitted between resources.

17. The process of claim 16 wherein the resources originate messages of ASCII text files and wherein the resources identify the system identity of messages from the text file.

18. The process of claim 17 wherein the text files contain a digital signature.

SELF CONFIGURING PEER TO PEER INTER PROCESS MESSAGING SYSTEM

ABSTRACT OF THE DISCLOSURE

The system provides remote program execution, data transport, message communication, status communication and relocation of computer resources by using an arbiter associated with each computer. An originating arbiter of a process resource sends messages between arbiters that are received by each arbiter and then sent to a destination arbiter, if required. If necessary, the message may be retransmitted by intermediate arbiters and eventually received by the destination arbiter which interpret, and executes the message. As a result, the arbiters provide actual communication between the resources. Each arbiter may be resident in each of a plurality of computers which are part of a network linked by a network. Each arbiter independently reviews and processes the messages so that the computers communicate directly with each other on a peer to peer basis without the need for a master controlling program or other gateway for controlling and processing the messages as the messages are transmitted between computers.

FIG. 1

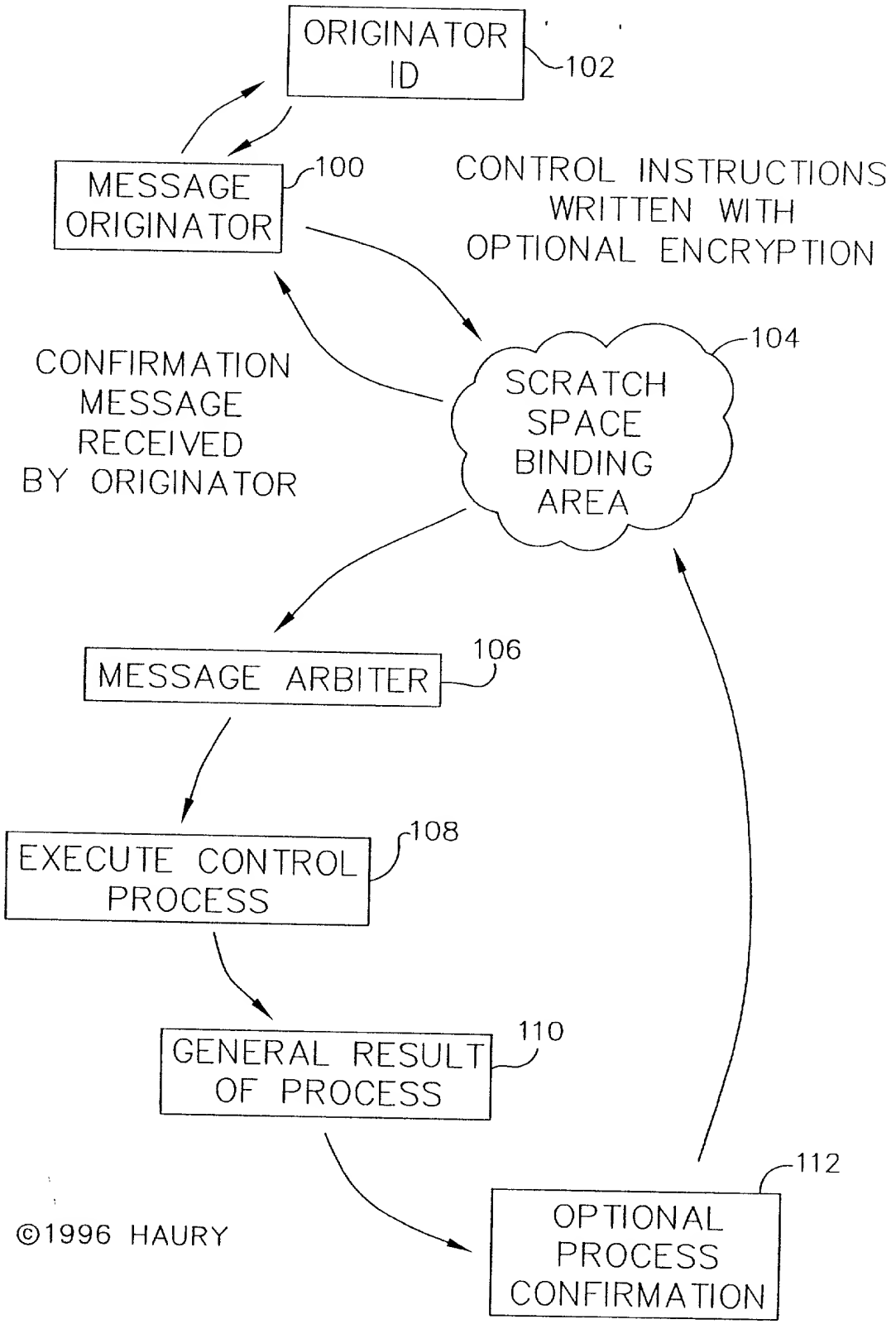


FIG. 2

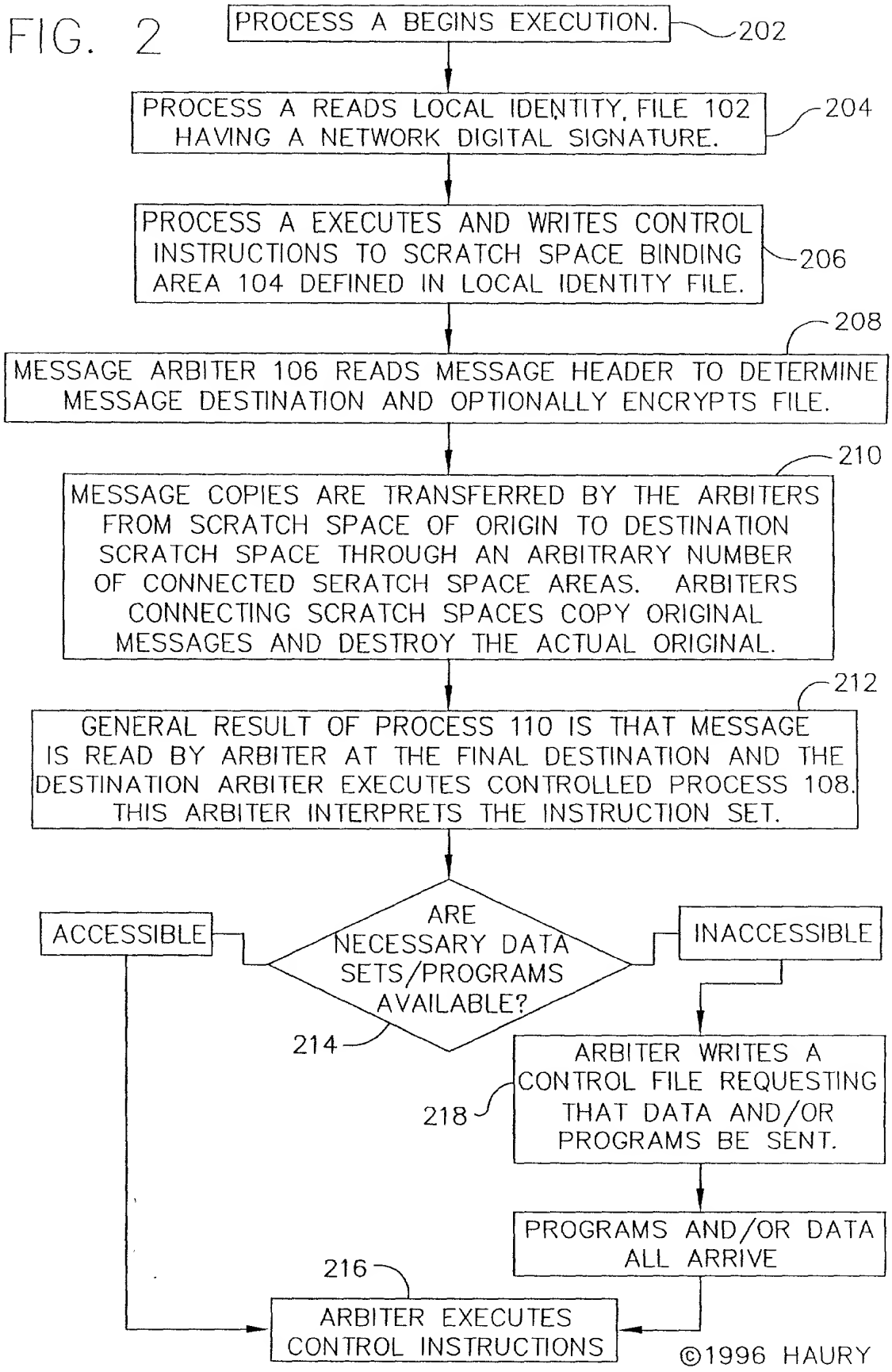

PROCESS A BEGINS EXECUTION.
202
PROCESS A READS LOCAL IDENTITY, FILE 102 HAVING A NETWORK DIGITAL SIGNATURE.
204
PROCESS A EXECUTES AND WRITES CONTROL INSTRUCTIONS TO SCRATCH SPACE BINDING AREA 104 DEFINED IN LOCAL IDENTITY FILE.
206
208
MESSAGE ARBITER 106 READS MESSAGE HEADER TO DETERMINE MESSAGE DESTINATION AND OPTIONALLY ENCRYPTS FILE.
210
MESSAGE COPIES ARE TRANSFERRED BY THE ARBITERS FROM SCRATCH SPACE OF ORIGIN TO DESTINATION SCRATCH SPACE THROUGH AN ARBITRARY NUMBER OF CONNECTED SERATCH SPACE AREAS. ARBITERS CONNECTING SCRATCH SPACES COPY ORIGINAL MESSAGES AND DESTROY THE ACTUAL ORIGINAL.
212
GENERAL RESULT OF PROCESS 110 IS THAT MESSAGE IS READ BY ARBITER AT THE FINAL DESTINATION AND THE DESTINATION ARBITER EXECUTES CONTROLLED PROCESS 108. THIS ARBITER INTERPRETS THE INSTRUCTION SET.
ARE NECESSARY DATA SETS/PROGRAMS AVAILABLE?
214
ACCESSIBLE
INACCESSIBLE
218
ARBITER WRITES A CONTROL FILE REQUESTING THAT DATA AND/OR PROGRAMS BE SENT.
PROGRAMS AND/OR DATA ALL ARRIVE
216
ARBITER EXECUTES CONTROL INSTRUCTIONS
©1996 HAURY

FIG. 3

ALTERNATE MESSAGE STRUCTURES

A (CONTEXT DEFINED)
FILENAME = PROCESS IDENTIFICATION
FILE EXTENSION = CONTROL INSTRUCTION
FILE CONTENTS = DATA SET AND/OR DATA POINTERS

B (CONTENT DEFINED)
ANY NUMBER OF LINES IN A ARBITRARY ORDER
SOURCE ID: (16 BYTES)
DESTINATION ID: (16 BYTES)
DATA SET: (AN ARBITRARY NUMBER OF BYTES)
PROGRAM: (AN ARBITRARY NUMBER OF BYTES)
SPECIAL INSTRUCTION: (AN ARBITRARY NUMBER OF BYTES); (...);...
KEYBOARD EXECUTION: (AN ARBITRARY NUMBER OF BYTES)
CONFIRMATION REQUEST: (1 BYTE)
RETURN ID: (16 BYTES)
RETURN DATA SET: (AN ARBITRARY NUMBER OF BYTES)
RETURN ENCRYPTION LEVEL: (1 BYTE)
NETWORK CONTROL: (16 BYTES)
DATE: (AN ARBITRARY NUMBER OF BYTES)
TIME: (AN ARBITRARY NUMBER OF BYTES)
SEQUENCE: (16 BYTES)

FIG. 4

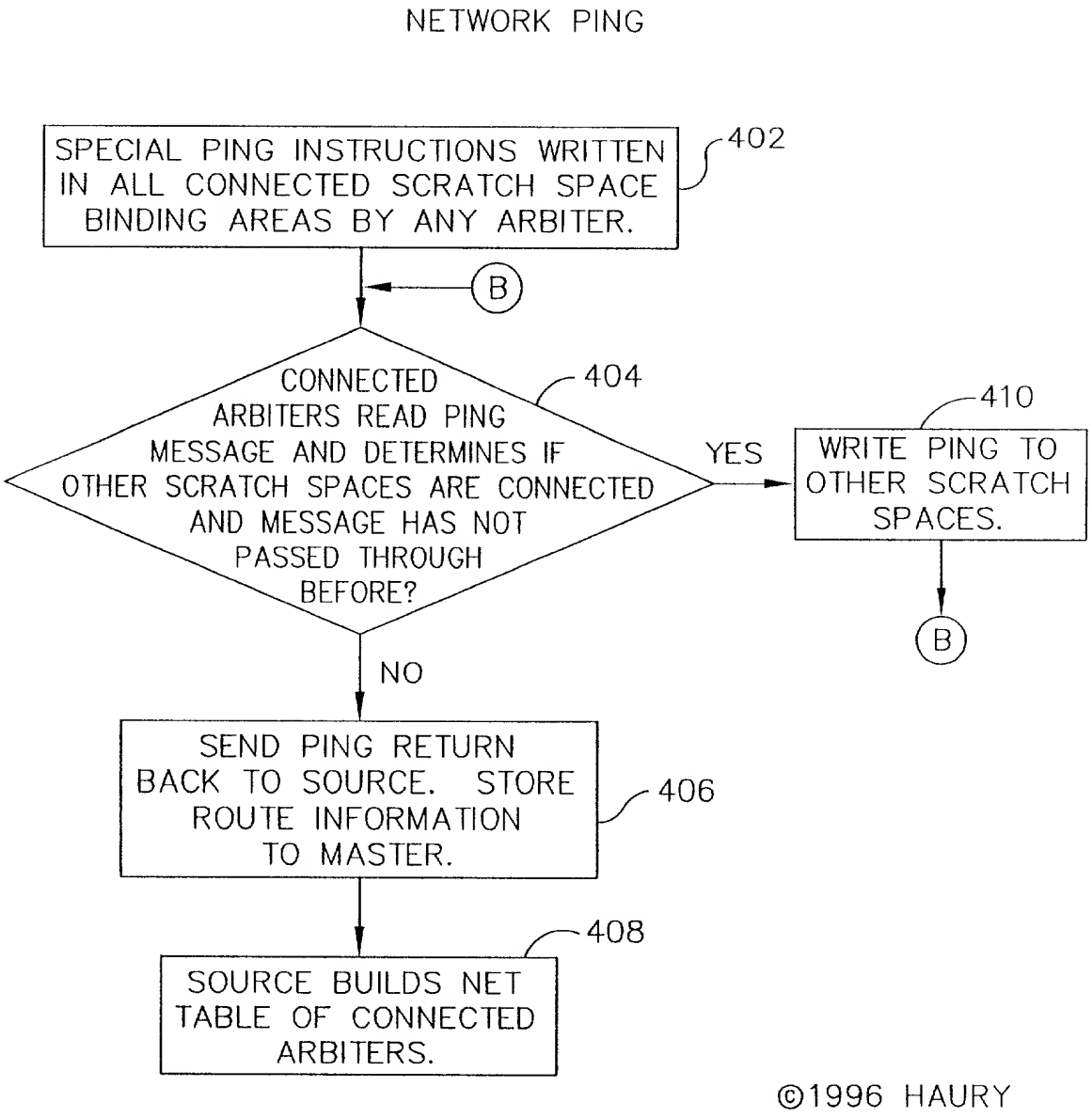


FIG. 5

PING MESSAGE

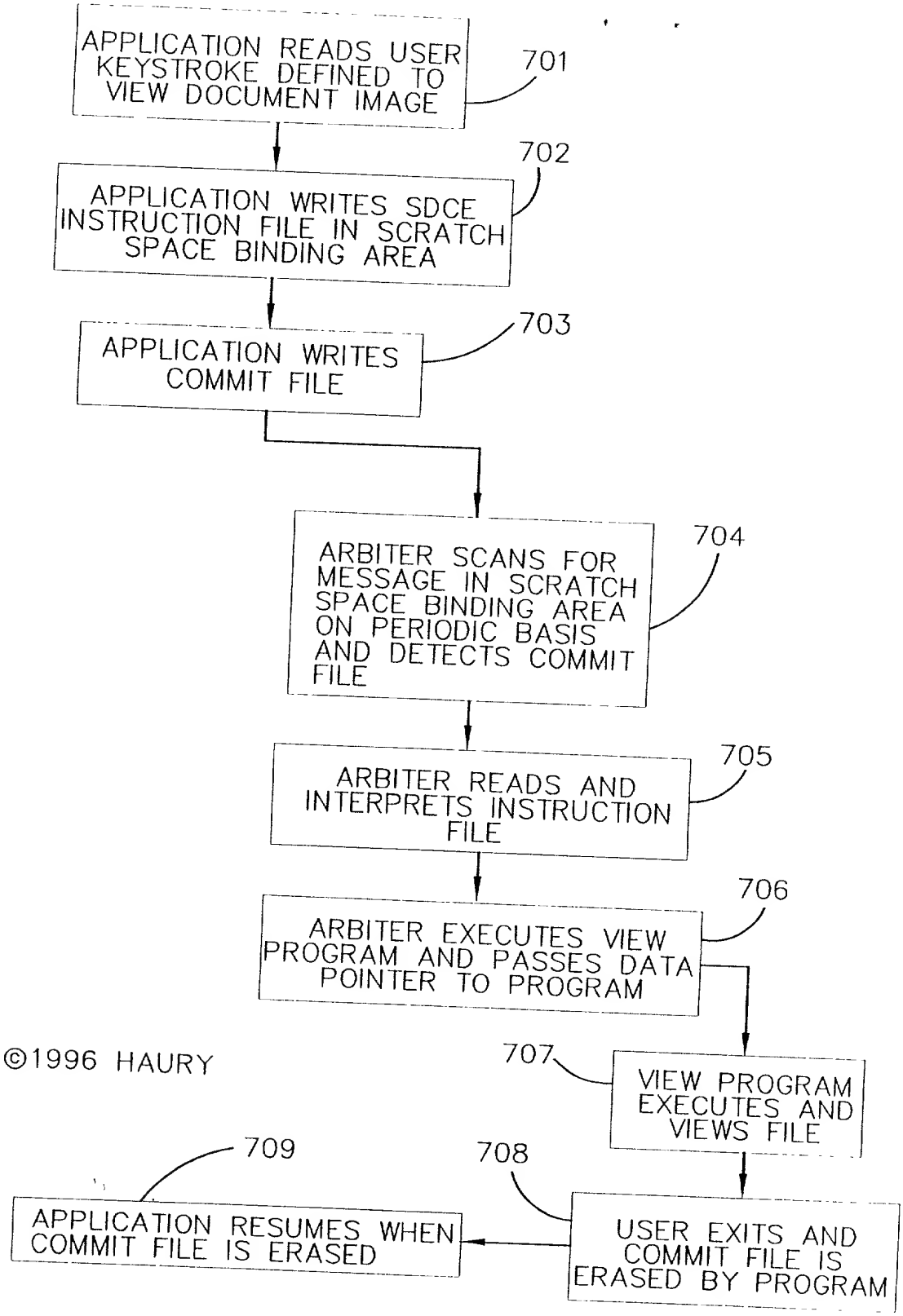
PING
(NODE ID #1) ORIGINATING NODE
(NODE ID #2) NODES ADDED
(NODE ID #3)
.
.
.
(NODE ID #n)

FIG. 6

CONTROL MESSAGE

1) TIME: (00: 00: 00: 00)
2) DATE: (xx/xx/xx)
3) RESET:
4) SILENCE:
5) KILL:
6) SHELL:
7) NEW MASTER: (XXXXXXXXXXXXXXX)
8) SLAVE MASTER: (XXXXXXXXXXXXXXX)
9) NET RESET
10) NET DOWN
11) CHANGE ID: (XXXXXXXXXXXXXXX)
12) HELLO
13) REQUEST NEW ID
14) MOVE DATA: (XXXXXX.XXX)
15) MOVE PROGRAM: (XXXXXX.XXX)
16) SEND MAP

FIG. 7



Attorney's Docket No. HWH 1821

DECLARATION AND POWER OF ATTORNEY

REGULAR OR DESIGN APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SELF CONFIGURING PEER TO PEER INTER PROCESS MESSAGING SYSTEM

the specification of which:

(check one)

[x] is attached hereto
[ ] was filed on _____ as Application Serial No. _____, and was amended on _____.
[ ] was described and claimed in PCT International Application No. _____, filed on _____ and as amended under PCT Article 19 on _____, if any.

ACKNOWLEDGEMENT OF REVIEW OF PAPERS AND DUTY OF CANDOR

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations §1.56.

PRIORITY CLAIM

I hereby claim foreign priority benefits under Title 35, United States Code, §119 (a) - (d) or §365(b) of any foreign application for patent or inventor's certificate, or §365(a) of any PCT application which designates at least one country other than the United States of America, listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Priority Claimed

(Number)	(Country)	(Day/Month/Year Filed)
(Number)	(Country)	(Day/Month/Year Filed)
(Number)	(Country)	(Day/Month/Year Filed)

Priority Not Claimed

ANY FOREIGN APPLICATION(S), ON THE SAME SUBJECT MATTER WHICH HAS A FILING DATE EARLIER THAN THE EARLIEST APPLICATION FROM WHICH PRIORITY IS CLAIMED

(Number)	(Country)	(Day/Month/Year Filed)

CLAIM FOR BENEFIT OF PROVISIONAL APPLICATION(S)

I hereby claim the benefit under Title 35, United States Code, §119(e) of any United States provisional application(s) listed below.

60/014,887 (Application Number)	4-5-96 (Filing Date)
(Application Number)	(Filing Date)

CLAIM FOR BENEFIT OF EARLIER U.S. APPLICATION(S) UNDER 35 U.S.C. 120

(complete this part only if this is a divisional, continuation or CIP application)

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s), or §365(c) of any PCT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code §112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application:

(Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)
(Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)

POWER OF ATTORNEY ,

I hereby appoint the following attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith.

Irving Powers (15,700), Donald G. Leavitt (17,626), John K. Roedel, Jr. (25,914), Michael E. Godar (28,416), Edward J. Hejlek (31,525), William E. Lahey (26,757), Richard G. Heywood (18,224), Frank R. Agovino (27,416), Kurt F. James (33,716), G. Harley Blosser (33,650), Paul I. J. Fleischut (35,513), Vincent M. Keil (36,838), Robert M. Evans, Jr. (36,794), Robert M. Bain (36,736), Kathleen M. Petrillo (35,076), Rudolph A. Telscher, Jr. (36,032), Paul A. Stone (38,628), Cindy S. Kaplan (40,043), David E. Crawford, Jr. (38,118), Paul A. Maddock (37,877), Charles E. Cohen (34,565), Scott A. Williams (39,876), and Richard L. Bridge (P-40529).

Send Correspondence To:

Customer Number: 000321

Direct Telephone Calls To:

Frank R. Agovino
(314) 231-5400

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

- -

Full name of sole or first inventor Harry R. Haury

Inventor's signature Harry R. Haury Date 4/4/97
Residence Chesterfield, Missouri Citizenship U.S.A.
Post Office address 801 Millfield Court
Chesterfield, Missouri 63017

Full name of second joint inventor _____

Second inventor's signature _____ Date _____
Residence _____ Citizenship _____
Post Office address _____